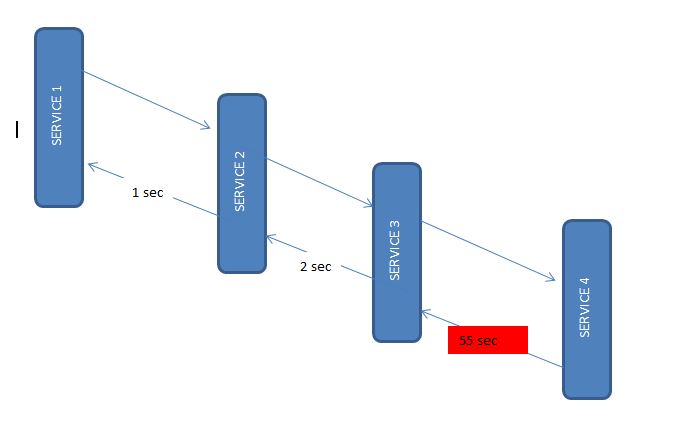
Lab 3 - Distributed tracing with Spring Cloud Sleuth and Zipkin

**Step 1: Set up the Project**

Let us create 4 spring boot based microservices. They all will have both Zipkin and Sleuth starter dependencies.

In each microservice, we will expose one endpoint and from the first service we will call the second service, and from the second service, we will invoke the third and so on using the RestTemplate.



Sleuth works automatically with resttemplate so it would send this instrumented service call information to the attached Zipkin server. Zipkin will then start the book keeping of latency calculation along with a few other statistics like service call details.

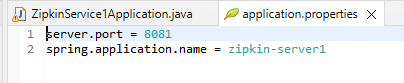
Here are the dependencies you need to include for each microservice.



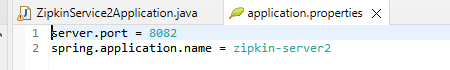
**Step 2: Instrument the Microservices**

Create multiple microservices using Spring Boot for demonstration purposes. Each microservice should be a Spring Boot application and have the necessary dependencies configured.

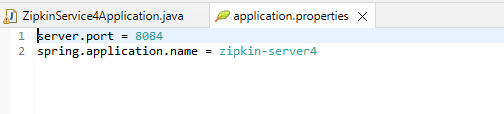
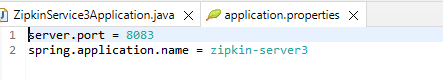




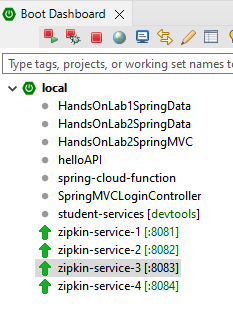






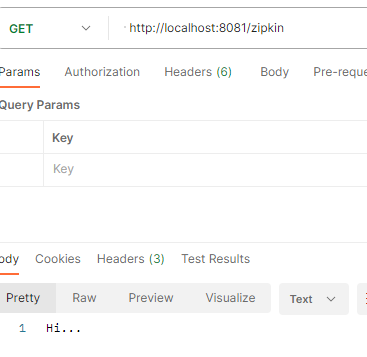


**Step 3: Start each microservice**



**Step 4: Test the Distributed Tracing**

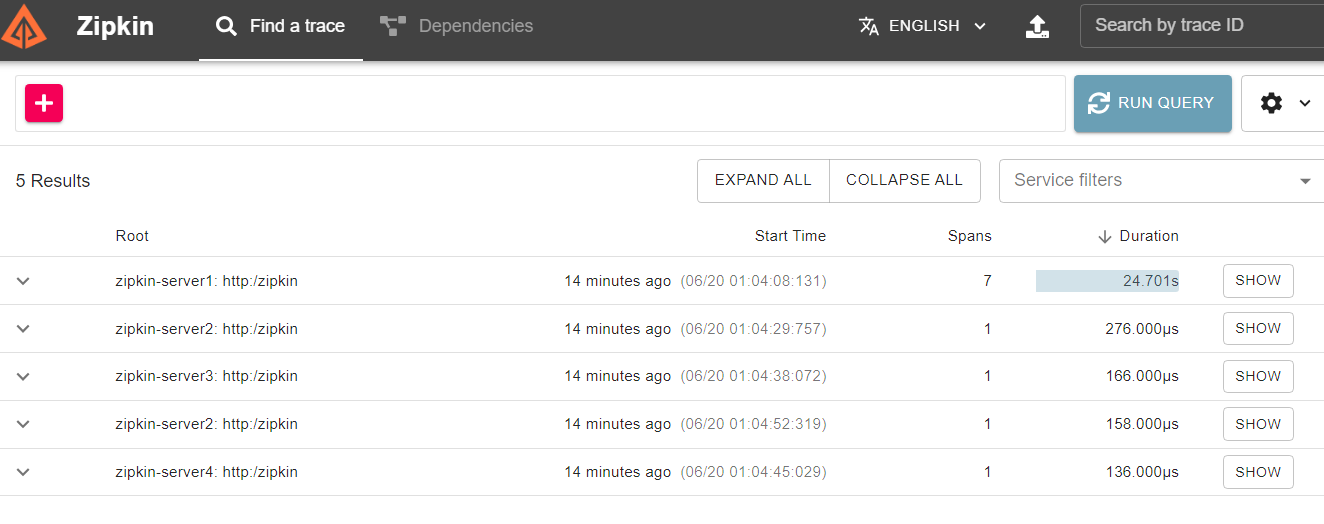
Make requests to trigger distributed tracing. Now test the first service endpoint couple of times from Postman – <http://localhost:8081/zipkin>.



Step 5: Visualize Traces

Access the web interface provided by the tracing system ie; Zipkin to view and analyze the captured traces.

Start up zipkin along with the Microservices.



This example demonstrates the basic setup and usage of Spring Cloud Sleuth for distributed tracing. In a real-world scenario, you would typically have more complex microservices and additional configurations based on your requirements.

Remember to adjust the dependencies and code based on your project's specific needs.

In this lab, we learned to use Zipkin to analyze latency in service calls. Also, we learned how Sleuth can help us create the metadata and pass it to Zipkin.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*